

Dependently Sorted Logic

João Filipe Belo

The University of Manchester, School of Computer Science
Oxford Road, Manchester, M13 9PL, UK
`beloj@cs.man.ac.uk`

Abstract. We propose syntax and semantics for systems of intuitionistic and classical first order *dependently sorted* logic, with and without equality, retaining type dependency, but otherwise abstracting, from systems for dependent type theory, and which can be seen as generalised systems of multisorted logic. These are presented as extensions to Gentzen systems for first order logic in which the logic is developed relative to a context of variable declarations over a theory of dependent sorts. A generalised notion of Kripke structure provides the semantics for the intuitionistic systems.

1 Introduction

Dependently sorted logic may be described as the generalisation of multisorted first order logic to a logic with dependent sorts, i.e., as the generalisation of multisorted first order logic in which the languages have been extended to include dependent sorts. This extension is nevertheless minimal in the sense that sort dependency is the only extra structure assumed.

We may thus describe the systems we introduce here as systems for multisorted first order logic generalised with dependent sorts. Alternatively, we may describe these systems as abstractions of *logic enriched type theories* [1] in which the only structure retained from the type theory is type dependency. The overall structure of these systems is thus that of a system of predicates and proofs over a type theory, or a *theory of sorts* [2].

The idea of minimally extending multisorted logic with dependent sorts has been addressed several times already [3,4,5,6] with seemingly different motivations. A system for intuitionistic dependently sorted logic without equality and an abstract notion of theory of dependent sorts called a *type setup* is proposed by Aczel in [3], starting the work we present in this paper. This is further developed in [4] with set theoretical semantics and completeness, but for classical logic instead. In [5], Makkai introduces a dependently sorted logic to formulate category theoretical notions, in which the use of equality is restricted, consequently avoiding function symbols. The paper [6] by Rabe is rather close in goal to ours but the development is somewhat different, deals only with classical logic, and imposes conditions on the syntax which, as said there, are rather restrictive, and which we don't need.

One motivation for developing dependently sorted logic as presented here is to fit dependent sorts in systems for software specification founded on multi-sorted logic, like CASL [7]. We want to try to apply these systems to generic programming following related work [8,9] in dependent type theory.

In this paper we propose syntax and semantics for systems of intuitionistic and classical dependently sorted logic, with and without equality, staying as close as possible to traditional presentations of multisorted logic. One may then be concerned with important properties of multisorted logic, like interpolation, still holding in these systems. In a subsequent paper we intend to show that Craig interpolation indeed holds for these systems.

This paper has two parts. In the first part we are concerned with theories of sorts. These are syntactical theories, so it begins with the notions of expression and substitution, after which the notion of signature is presented. Signatures are the simplest of the theories of sorts due to the absence of equality. This is followed by the notion of generalised algebraic theory, introduced by Cartmell [10,11,12], which adds equality both on terms and on sorts. Then set theoretical semantics of generalised algebraic theories is given and a completeness result is proved. The second part presents the notion of a dependently sorted first order theory and its complete semantics, both classical and intuitionistic.

2 Theories of Dependent Sorts

Roughly, a theory of dependent sorts is one which allows *individual* variables to occur in the expression of sorts. The actual sort, in the multisorted sense, *depends* on the value of the variables. It's very much like the truth value of a predicate depending on the value of the variables occurring in it. Consider, for instance, the expression $vec(n)$ for *the sort of vectors of length n* [13].

Sort dependency is the only extra structure, with regard to sorts in the multisorted sense, that we assume in dependently sorted logic. We ignore further structure, like dependent products, common in dependent type theories.

This section presents two theories of dependent sorts, namely signatures and generalised algebraic theories. The distinguishing feature among them is equality, which is absent in signatures. The main observation about the presentation of these theories is that the inductive definition of sorts and terms must be simultaneous by the nature of dependent sorts.

2.1 Expressions

We shall now give a definition of a notion of expression, which we do mainly to simplify the definition of substitution on the objects we shall introduce later. We give it in very much the standard way where the expressions are built from variables by a series of symbol applications. It should nevertheless be noted that for now we don't assign arities to the symbols, and hence that we don't impose any arity constraint on the formation of expressions. It should also be noted that we intend a notion of expression modulo renaming of bound variables, or *α -conversion*.

For certain inductively defined sets we need an objective or formal notion of *derivation* according to the clauses of the inductive definition. These sets shall be defined indirectly through the inductive definition of the derivations instead. For this we use the notation

$$\frac{\Pi_1 \quad \dots \quad \Pi_k}{Q} \{ \mathcal{R} \},$$

to abbreviate a clause “If Π_1, \dots, Π_k are derivations, then (Π_1, \dots, Π_k, Q) is a derivation, provided that \mathcal{R} .” We may omit \mathcal{R} , which we then consider to hold. The Π_i we call the *premises* of the derivation and Q the *conclusion*. When writing down such a clause we shall in fact, of each Π_i , show only its conclusion. We say that “ Q is derivable,” write $\vdash Q$, when Q is the conclusion of some derivation. The sets indirectly being defined are the sets of those Q which are derivable. For uniformity sake, hereafter we shall use this notation in every inductive definition, even when we don’t need the formal derivations.

For us a sequence is always a possibly empty finite sequence, the empty sequence being denoted by the letter ϵ . In a situation where a sequence is specified, say E_1, \dots, E_n , the sequence may be denoted simply by the unsubscripted letter, say E .

Definition 1. We assume fixed an infinite set U of symbols and an infinite set V of variables and inductively define the expressions by the following clauses.

Variable, compound, variable binding

$$\frac{}{v} \{v \in V\} \quad \frac{E_1 \quad \dots \quad E_n}{H(E)} \{H \in U\} \quad \frac{A \quad \psi}{(Hv : A)\psi} \{H \in U \text{ and } v \in V\}$$

We may use infix notation $E_1 H E_2$ when denoting expressions of the form $H(E_1, E_2)$ and also omit the parenthesis when denoting expressions of the form $H()$. Moreover, in this subsection we let the letters, possibly subscripted or primed,

- H denote an arbitrary symbol,
- u, v, w, x, y denote arbitrary variables,
- A, C, D, E, ψ denote arbitrary expressions,

except otherwise indicated.

Definition 2. An occurrence of a variable v in an expression E is free (bound) in E according to the following induction on the structure of E : (1) the occurrence of v in v is free; (2) a free (bound) occurrence of v in E_i is free (bound) in $H(E_1, \dots, E_n)$; (3) a free (bound) occurrence of v in A is free (bound) in $(Hv : A)\psi$; (4) a free (bound) occurrence of v in ψ is free (bound) in $(Hu : A)\psi$ if v is distinct from u , otherwise is bound.

Definition 3. An occurrence of a variable v in an expression E is said to be binding in E according to the following induction on the structure of E : (1) the occurrence of v in v is not binding; (2) a binding occurrence of v in E_i is binding in $H(E_1, \dots, E_n)$; (3) the occurrence of v in $(Hv : A)\psi$ is binding; (4) a binding occurrence of v in ψ is binding in $(Hu : A)\psi$.

Definition 4. A variable v is said to be free in an expression E if v has a free occurrence in E and v is said bound in E if v has a binding occurrence in E . The set $\text{FV}(E)$ is the set of free variables in E . The set $\text{FV}(E_1, \dots, E_n)$ is the union of the set of free variables of each E_i .

Definition 5. A substitution is a pair of sequences $D_1, \dots, D_m/y_1, \dots, y_m$, for any $m \geq 0$, such that the variables in y are distinct.

Definition 6. Let the pair $D_1, \dots, D_m/y_1, \dots, y_m$ be a substitution. The simultaneous substitution $E[D/y]$ of D for y in E is defined by induction on the structure of E by equations:

$$\begin{aligned} v[D/y] &= D_i && \text{if } v = y_i \text{ for some } i, \\ v[D/y] &= v && \text{if } v \neq y_i \text{ for all } i, \\ H(E_1, \dots, E_n)[D/y] &= H(E_1[D/y], \dots, E_n[D/y]) \\ (Hv : A)\psi[D/y] &= (Hv : A)(\psi[D'/y']) && \text{if } v \notin \text{FV}(D), \end{aligned}$$

where D'/y' is

$$\begin{aligned} D_1, \dots, D_{i-1}, D_{i+1}, \dots, D_m/y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_m & \text{ if } v = y_i \text{ for some } i, \\ D/y & \text{ if } v \neq y_i \text{ for all } i. \end{aligned}$$

Each equation must be considered under the condition that the simultaneous substitutions on its right hand side are defined.

Proposition 1 ((Strict) Substitution lemma). Let $C_1, \dots, C_l/x_1, \dots, x_l$ and $D_1, \dots, D_m/y_1, \dots, y_m$ be substitutions. Then

$$E[C/x][D/y] = E[D/y][C[D/y]/x],$$

provided the simultaneous substitutions are defined and none of the variables in x is in y or occurs free in D .

As said, we intend a notion of expression modulo renaming of bound variables. For that we define *syntactic equality* which relates those expressions differing only in bound variables.

Definition 7. Syntactic equality \equiv of expressions is inductively defined by the clauses:

Reflexivity, congruence, bound variable renaming

$$\frac{}{v \equiv v} \quad \frac{E_1 \equiv E'_1 \quad \dots \quad E_n \equiv E'_n}{H(E) \equiv H(E')} \quad \frac{\psi[w/v] \equiv \psi'[w/v']}{(Hv : A)\psi \equiv (Hv' : A)\psi'}$$

where w is some variable occurring neither in ψ nor in ψ' .

Proposition 2. Syntactic equality is an equivalence relation.

Proposition 3. *Let E and E' be expressions such that $E \equiv E'$. Then*

$$E[D/y] \equiv E'[D/y],$$

for any substitution $D_1, \dots, D_m/y_1, \dots, y_m$ for which the simultaneous substitutions are defined.

Proposition 4. *For any expression E and substitution $D_1, \dots, D_m/y_1, \dots, y_m$ there exists an expression E' such that $E \equiv E'$ and $E'[D/y]$ is defined.*

Proposition 4 implies that simultaneous substitution on equivalence classes of expressions is totally defined. Thus, hereafter the expressions are taken modulo syntactic equality, or renaming of bound variables.

2.2 Signatures

The simplest way to set up a theory of sorts with sort dependency is, perhaps, through the notion of signature we present in this section. The fundamental notions in this presentation are those of *context*, of *sort*, and of *term*. The set up will be such that (1) every sort and term shall explicitly contain a context declaring the variables from which it may be formed: no variable may be used other than those declared in that context; and (2) every term shall explicitly contain an expression designating its sort.

Definition 8. *A variable declaration is a pair $v : A$, where v is a variable and A is an expression.*

Signatures enjoy the relevant properties of the generalised algebraic theories we present in the next section. In fact they are a particular kind of generalised algebraic theory. We thus choose to develop signatures only to the point where we can give some examples involving dependent sorts.

As metavariables, we now let the letters, possibly subscripted or primed,

- f, g, h, F, G denote arbitrary symbols,
- u, v, w denote arbitrary variables,
- $p, q, r, s, t, A, B, C, D$ denote arbitrary expressions,
- Γ, Δ, E denote arbitrary sequences of variable declarations,

except otherwise indicated. Also, unless stated otherwise, we assume the sequences of variables in order declared in Γ and Δ to be x_1, \dots, x_m and y_1, \dots, y_n , respectively.

Definition 9. *We start with the following definitions.*

1. *A sort constructor declaration is a pair $(\Delta) F$.*
2. *A term constructor declaration is a triple $f :: \Delta \rightarrow D$.*

Sort and term constructor declarations simply assign arities to symbols: the sequence of variable declarations assigned to a symbol determines the number

of terms, and their sort, to which that symbol may be applied, and the target expression of a term constructor declaration determines furthermore the sort of that application. Note nevertheless that, at this point, symbols are assigned arbitrary sequences of variable declarations, i.e., the expressions assigned to the variables in those sequences don't necessarily designate sorts. Nor for that matter does the target expression of a term constructor declaration designate a sort, as it should. These two constraints must be further imposed by stating that, in a proper collection of declarations, in what we shall call a signature, the sequences of variable declarations assigned to the symbols must actually be contexts and, roughly, the target expressions of term constructor declarations must be sorts.

Definition 10. A signature is a set Σ of sort and term constructor declarations such that

1. there are no two declarations for the same symbol in Σ ,
2. if $(\Delta) F \in \Sigma$, then $\vdash \Delta$, and
3. if $f :: \Delta \rightarrow D \in \Sigma$, then $\vdash (\Delta) D$,

according to the following clauses for the simultaneous inductive definition of contexts Γ , sorts $(\Gamma) A$, and terms $(\Gamma) t : A$ over Σ , where $(\Gamma) t : \Delta$ abbreviates the list of premises

$$\Gamma \quad (\Gamma) t_1 : B_1 \quad \dots \quad (\Gamma) t_n : B_n[t_1, \dots, t_{n-1}/y_1, \dots, y_{n-1}] \quad \Delta$$

for B_1, \dots, B_n the sequence of sorts of the variables in order declared in Δ . Note that since Δ may be the empty sequence, the premise Γ is necessary to disallow an arbitrary sequence of variables as the context of the conclusion.

empty context, context extension

$$\frac{}{\epsilon} \quad \frac{(\Gamma) A}{\Gamma, v : A} \{v \notin \Gamma\}$$

sort constructor application

$$\frac{(\Gamma) t : \Delta}{(\Gamma) F(t)} \{(\Delta) F \in \Sigma\}$$

declared variable, term constructor application

$$\frac{\Gamma}{(\Gamma) v : A} \{v : A \in \Gamma\} \quad \frac{(\Gamma) t : \Delta \quad (\Delta) D}{(\Gamma) f(t) : D[t/y]} \{f :: \Delta \rightarrow D \in \Sigma\}$$

We may omit the pair of parenthesis when denoting an application of a constructor symbol to the empty sequence. We usually omit the context, and the arrow, in the declaration of sort and term constructors when the context is empty.

Example 1. The canonical example of a signature is perhaps that for a theory of categories, where the set of morphisms between two objects is designated by a dependent sort:

Sort constructors

$() \text{ Obj}$
 $(x : \text{Obj}, y : \text{Obj}) \text{ Arr}$

Term constructors

$\circ :: (x : \text{Obj}, y : \text{Obj}, z : \text{Obj}, g : \text{Arr}(x, y), f : \text{Arr}(y, z)) \rightarrow \text{Arr}(x, z)$
 $\text{id} :: (x : \text{Obj}) \rightarrow \text{Arr}(x, x)$

Example 2. A more interesting example deals with indexed families of categories:

Sort constructors

$() \text{ V}$
 $(i : \text{V}) \text{ Obj}$
 $(i : \text{V}, x : \text{Obj}(i), y : \text{Obj}(i)) \text{ Arr}$

Term constructors

$\circ :: (i : \text{V}, x : \text{Obj}(i), \dots f : \text{Arr}(i, y, z)) \rightarrow \text{Arr}(i, x, z)$
 $\text{id} :: (i : \text{V}, x : \text{Obj}(i)) \rightarrow \text{Arr}(i, x, x)$

Example 3. Another example is that of a signature for a theory of stacks, where the sort of a stack includes the number of elements in it:

Sort constructors

$() \text{ Nat}$
 $() \text{ T}$
 $(\text{len} : \text{Nat}) \text{ Stk}$

Term constructors

$0 :: \text{Nat}$
 $\text{suc} :: \text{Nat} \rightarrow \text{Nat}$
 $\text{empty} :: \text{Stk}(0)$
 $\text{push} :: (\text{len} : \text{Nat}, s : \text{Stk}(\text{len}), e : \text{T}) \rightarrow \text{Stk}(\text{suc}(\text{len}))$
 $\text{top} :: (\text{len} : \text{Nat}, s : \text{Stk}(\text{suc}(\text{len}))) \rightarrow \text{T}$

This disallows, for instance, the application of *top* to the *empty* stack.

Before we move on to the next subject we note, regarding the definition of signature, that dropping the premise Δ from the sort constructor application clause, and the premises Δ and $(\Delta) D$ from the term constructor application clause, enables the following to be a signature.

Sort constructors

$(x : G(c)) G$

Term constructors

$c :: G(c)$

Note the “circularity” in the declarations. The theory can still be developed in this case, without much change, although the proofs are not so straightforward since the language no longer has a proper simultaneous inductive definition.

2.3 Generalised Algebraic Theories

The notion of generalised algebraic theory extends that of signature with the expression of equality both on terms and on sorts. It was introduced by Cartmell [10] as “a generalisation of the usual notion of a many-sorted algebraic or equational theory.” Our presentation uses slightly different notation and terminology in a style closer to that of the traditional presentations of multisorted languages.

The key observation here is that an inferred equality may enable a new application of a constructor symbol, thus the contexts, the sorts, and the terms must be formed simultaneously with the inference of equality.

Definition 11

1. An algebraic equation axiom is a triple $(\Gamma) r = s : A$.
2. A sort equation axiom is a triple $(\Gamma) A = B$.

Definition 12. A generalised algebraic theory is a set Σ of sort constructor declarations, term constructor declarations, algebraic equation axioms, and sort equation axioms such that

1. there are no two declarations for the same symbol in Σ ,
2. if $(\Delta) F \in \Sigma$, then $\vdash \Delta$,
3. if $f :: \Delta \rightarrow D \in \Sigma$, then $\vdash (\Delta) D$,
4. if $(\Gamma) s = t : A \in \Sigma$, then $\vdash (\Gamma) s : A$ and $\vdash (\Gamma) t : A$,
5. if $(\Gamma) A = B \in \Sigma$, then $\vdash (\Gamma) A$ and $\vdash (\Gamma) B$,

according to the following clauses for the simultaneous inductive definition of contexts Γ , sorts $(\Gamma) A$, terms $(\Gamma) t : A$, algebraic equations $(\Gamma) r = s : A$, and sort equations $(\Gamma) A = B$ over Σ . For brevity we omit the signature clauses for the contexts, sorts, and terms.

sort replacement on terms

$$\frac{(\Gamma) r : A \quad (\Gamma) A = B}{(\Gamma) r : B}$$

algebraic equation axiom

$$\frac{(\Gamma) r : A \quad (\Gamma) s : A}{(\Gamma) r = s : A} \{(\Gamma) r = s : A \in \Sigma\}$$

reflexivity, symmetry, transitivity

$$\frac{(\Gamma) t : A}{(\Gamma) t = t : A} \quad \frac{(\Gamma) r = s : A}{(\Gamma) s = r : A} \quad \frac{(\Gamma) r = s : A \quad (\Gamma) s = t : A}{(\Gamma) r = t : A}$$

substitution on algebraic equations

$$\frac{(\Gamma) r = s : \Delta \quad (\Delta) p = q : D}{(\Gamma) p[r/y] = q[s/y] : D[r/y]}$$

sort replacement on algebraic equations

$$\frac{(\Gamma) r = s : A \quad (\Gamma) A = B}{(\Gamma) r = s : B}$$

sort equation axiom

$$\frac{(\Gamma) A \quad (\Gamma) B}{(\Gamma) A = B} \{(\Gamma) A = B \in \Sigma\}$$

reflexivity, symmetry, transitivity of sort equations

$$\frac{(\Gamma) A}{(\Gamma) A = A} \quad \frac{(\Gamma) A = B}{(\Gamma) B = A} \quad \frac{(\Gamma) A = B \quad (\Gamma) B = C}{(\Gamma) A = C}$$

substitution on sort equations

$$\frac{(\Gamma) r = s : \Delta \quad (\Delta) C = D}{(\Gamma) C[r/y] = D[s/y]}$$

We next display a series of fundamental properties of these derivations, writing $\vdash (\Gamma) t : \Delta$ as an abbreviation for

$$\vdash \Gamma, \vdash (\Gamma) t_1 : B_1, \dots, \vdash (\Gamma) t_n : B_n[t_1, \dots, t_{n-1}/y_1, \dots, y_{n-1}], \text{ and } \vdash \Delta,$$

for Δ the sequence $y_1 : B_1, \dots, y_n : B_n$ of variable declarations.

Proposition 5. *Let Σ be a generalised algebraic theory.*

- i. *If u occurs in p in $\vdash (\Delta) p : D$, then $\vdash (\Delta) u : B_i$ for some i .*
- ii. *If u occurs in D in $\vdash (\Delta) D$, then $\vdash (\Delta) u : B_i$ for some i .*

Proposition 6. *Suppose that $\vdash (\Gamma) t : \Delta$.*

- i. *If $\vdash (\Delta) p : D$, then $\vdash (\Gamma) p[t/y] : D[t/y]$,*
- ii. *If $\vdash (\Delta) D$, then $\vdash (\Gamma) D[t/y]$.*

Proposition 7

- i. *If $\vdash (\Gamma) A$, then $\vdash \Gamma$.*
- ii. *If $\vdash \Gamma$ and $v : A \in \Gamma$, then $\vdash (\Gamma) A$.*
- iii. *If $\vdash (\Gamma) r = s : A$, then $\vdash (\Gamma) r : A$ and $\vdash (\Gamma) s : A$.*
- iv. *If $\vdash (\Gamma) A = B$, then $\vdash (\Gamma) A$ and $\vdash (\Gamma) B$.*
- v. *If $\vdash (\Gamma) r : A$, then $\vdash (\Gamma) A$.*

Proposition 8. *If $\vdash (\Gamma) r : A$ and $\vdash (\Gamma) r : B$, then $\vdash (\Gamma) A = B$.*

Proof. The proof is by a double induction on the height of the derivations. Take a derivation of $\vdash (\Gamma) r : A$ and a derivation of $\vdash (\Gamma) r : B$. They must be a variable, a term constructor application, or a sort replacement on terms. For every possible combination of these it can be checked that $\vdash (\Gamma) A = B$ indeed holds either by reflexivity or by the hypothesis of induction and transitivity.

3 Dependently Sorted Algebras

We now turn to the set theoretical semantics for generalised algebraic theories which we prove sound and complete, although in a weak sense. The semantics is based on the idea that a sort is interpreted by a family of sets indexed by the interpretation of its context, and that a term is interpreted by a function on the interpretation of its context to the interpretation of its sort, but in a way which respects the indexing of that sort. The condition that such an interpretation respects the interpretation of substitution as composition defines the notion of structure for a generalised algebraic theory. If furthermore the equality axioms are satisfied by the interpretation, then the structure is called a dependently sorted algebra. Thus, let Σ be a generalised algebraic theory.

Definition 13. A structure M for Σ is a triple of interpreting functions $\llbracket \cdot \rrbracket_M$ on contexts, sorts, and terms such that

1. $\llbracket \Gamma \rrbracket_M$ is a set
 $\llbracket () \rrbracket_M = \{\emptyset\}$
 $\llbracket \Gamma, v : A \rrbracket_M = \{(e, e') \mid e \in \llbracket \Gamma \rrbracket_M \text{ and } e' \in \llbracket (\Gamma) A \rrbracket_M(e)\}$
2. $\llbracket (\Gamma) A \rrbracket_M$ is a family of sets indexed by $\llbracket \Gamma \rrbracket_M$
 $\llbracket (\Gamma) D[t/y] \rrbracket_M(e) = \llbracket (\Delta) D \rrbracket_M(\llbracket (\Gamma) t : \Delta \rrbracket_M(e))$
3. $\llbracket (\Gamma) r : A \rrbracket_M : (e \in \llbracket \Gamma \rrbracket_M) \rightarrow \llbracket (\Gamma) A \rrbracket_M(e)$
 $\llbracket (\Gamma) x_i : A_i \rrbracket_M(e) = e_i$
 $\llbracket (\Gamma) p[t/y] : D[s/y] \rrbracket_M(e) = \llbracket (\Delta) p : D \rrbracket_M(\llbracket (\Gamma) t : \Delta \rrbracket_M(e))$

where $x_i : A_i$ is the i th variable declaration in Γ and $\llbracket (\Gamma) t : \Delta \rrbracket_M(e)$ abbreviates

$$(\llbracket (\Gamma) t_1 : B_1 \rrbracket_M(e), \dots, \llbracket (\Gamma) t_n : B_n[t_1, \dots, t_{n-1}/y_1, \dots, y_{n-1}] \rrbracket_M(e)),$$

B_1, \dots, B_n being the sequence of sorts of the variables in order declared in Δ .

One can also define a structure to be an assignment of a family of sets to each sort constructor and of a function to each term constructor such that the above equations, slightly changed and taken in general as inductively defining a partial assignment on the contexts, sorts, and terms, are indeed total. Nevertheless, the above definition captures the properties needed for what follows.

Definition 14. An algebraic equation $(\Gamma) r = s : A$ is valid in a structure M for Σ , written $M \models_{\Sigma} (\Gamma) r = s : A$, if $\llbracket (\Gamma) r : A \rrbracket_M = \llbracket (\Gamma) s : A \rrbracket_M$. Similarly for a sort equation. The structure M is called a (dependently sorted) algebra for Σ if every algebraic and sort equation in Σ is valid in M .

Proposition 9 (Soundness)

i. For any algebraic equation $(\Gamma) r = s : A$ over Σ ,

$$\vdash (\Gamma) r = s : A \text{ only if } M \models_{\Sigma} (\Gamma) r = s : A \text{ for any algebra } M \text{ for } \Sigma.$$

ii. For any sort equation $(\Gamma) A = B$ over Σ ,

$\vdash (\Gamma) A = B$ only if $M \models_{\Sigma} (\Gamma) A = B$ for any algebra M for Σ .

Proof By induction on the height of the derivation of $(\Gamma) r = s : A$ and $(\Gamma) A = B$, using properties 2 and 3 of the definition of structure for Σ in the case of a substitution on sort equations and substitution on algebraic equations, respectively.

3.1 Completeness

By completeness we understand the converse of propositions 9.i and 9.ii. We shall show that the converse of 9.ii does not hold, so we aim solely at the converse of 9.i, which we call *weak* completeness. The method we use is the standard one of building a term algebra, and then showing that it only validates a given algebraic equation if that equation is derivable.

Thus, let Σ be a generalised algebraic theory. We shall prove the following proposition.

Proposition 10 (Weak Completeness). *For any equation $(\Gamma) r = s : A$, it holds that $\vdash (\Gamma) r = s : A$ if $M \models_{\Sigma} (\Gamma) r = s : A$ for any algebra M .*

For simplicity of notation, we restrict the proof to the case where Γ is the empty context, and thus build the term algebra by collecting closed terms. For the general case one would collect terms in the context Γ instead.

Definition 15. *We say a sort A is closed if the sort has the empty context, and similarly for terms. We'll often leave out the empty context when referring to closed sorts and closed terms.*

Recall proposition 8 for the next definition.

Proposition 11. *Let the binary relation \simeq on the closed terms over Σ be defined by $r : A \simeq s : B$ if $\vdash r = s : B$, for closed terms $r : A$ and $s : B$. Then,*

- i. \simeq is an equivalence relation,
- ii. $c : \Gamma' \simeq e : \Gamma'$ implies $r : A[c/x] \simeq r : A[e/x]$, for $\vdash (\Gamma') A$ and $\vdash r : A[e/x]$,
and
- iii. $c : \Gamma' \simeq e : \Gamma'$ implies $t[c/x] : A[c/x] \simeq t[e/x] : A[e/x]$, for $\vdash (\Gamma') t : A$.

Definition 16. *The canonical structure for Σ is the triple of assignments $\llbracket \cdot \rrbracket$ defined on the contexts Γ' , sorts $(\Gamma') A$, and terms $(\Gamma') t : A$ by an induction on the number of declarations in Γ' ,*

1. $\llbracket () \rrbracket = \{\emptyset\}$,
2. $\llbracket \Gamma', y : B \rrbracket = \{(e, e') \mid e \in \llbracket \Gamma' \rrbracket \text{ and } e' \in \llbracket (\Gamma') B \rrbracket(e)\}$,
3. $\llbracket (\Gamma') A \rrbracket$ is a family of sets indexed by $\llbracket \Gamma' \rrbracket$
 $\llbracket (\Gamma') A \rrbracket([e]_{\simeq}) = \{[r : A[e/x]]_{\simeq} \mid \vdash r : A[e/x]\}$,
4. $\llbracket (\Gamma') t : A \rrbracket : ([e]_{\simeq} \in \llbracket \Gamma' \rrbracket) \rightarrow \llbracket (\Gamma') A \rrbracket([e]_{\simeq})$

$$\llbracket (\Gamma') t : A \rrbracket ([e]_{\simeq}) = [t[e/x] : A[e/x]]_{\simeq},$$

where x is the sequence of variables in order declared in Γ' .

Proposition 12. *The canonical structure for Σ is a structure for Γ' .*

Proof. The proof is by induction on the number of variables declared in Γ' , checking the conditions in the definition of structure. It essentially follows from the properties of substitution.

Proposition 13. *The canonical structure for Σ is an algebra for Σ .*

Proof. It needs only to be checked that, whenever $(\Gamma') r = s : A \in \Sigma$,

$$\llbracket (\Gamma') r : A \rrbracket ([e]_{\simeq}) = \llbracket (\Gamma') s : A \rrbracket ([e]_{\simeq}), \text{ for all } [e]_{\simeq} \in \llbracket \Gamma' \rrbracket,$$

and that whenever $(\Gamma') A = B \in \Sigma$,

$$\llbracket (\Gamma') A \rrbracket ([e]_{\simeq}) = \llbracket (\Gamma') B \rrbracket ([e]_{\simeq}), \text{ for all } [e]_{\simeq} \in \llbracket \Gamma' \rrbracket.$$

Proposition 14. *Let Σ be a generalised algebraic theory, and let M be the canonical structure for Σ . Then,*

$$M \models_{\Sigma} r = s : A \text{ only if } \vdash r = s : A,$$

for closed terms $r : A$ and $s : A$.

Proof. This is an immediate consequence of the definition of \simeq .

Again, we don't claim that $M \models_{\Sigma} A = B$ only if $\vdash A = B$, for closed sorts A and B . Here a problem arises as follows. Suppose Σ is the generalised algebraic theory $\{() A, () B, (x : A) A = B, (x : B) A = B\}$. Then $\not\vdash A = B$, but $M \models_{\Sigma} A = B$ for every algebra M for Σ . This is because in every algebra either A and B are interpreted by the empty set, and then their interpretations are equal, or must otherwise be interpreted by equal sets as imposed by the axioms. Thus we have the following.

Proposition 15. *There is a generalised algebraic theory over which an equality on sorts is not derivable but is nevertheless satisfied in every algebra.*

Thus, for the stronger completeness result we need a more general category, one which allows a more intensional notion of equality between the objects interpreting the sorts.

4 Systems of Dependently Sorted Logic

As said in the introduction, we consider a logic system to be a system of predicates and proofs fitted over a theory of sorts. The purpose of this section is to make precise what we mean by this for the particular case of dependently sorted logic.

We shall introduce systems for both classical and intuitionistic first order logic. The inference rules are essentially those of the systems G1c and G1i in [14], except that:

1. the sequents in our systems include a context, which is a way of dealing with empty sorts as the availability of variables of given sorts is implicitly an existence assumption, see [15], page 811, and
2. a substitution rule is included with an algebraic equality as a side condition, so that a term in the conclusion of a derivation may be replaced by another algebraically equal to it.

We use the letters, possibly subscripted or primed, ζ , ρ , ϕ , ψ to denote arbitrary expressions, and Θ , P , Φ to denote arbitrary multisets of expressions.

Definition 17

1. A predicate symbol declaration is a pair $R \subset (\Delta)$.
2. A logic system is a generalised algebraic theory Σ together with a set of predicate symbol declarations $R \subset (\Delta)$ such that $\vdash_{\Sigma} \Delta$.

Let Σ be a logic system. We shall use the same letter to denote a logic system and its theory of sorts.

Definition 18. *The formulas over Σ are inductively defined according to the following clauses.*

atomic, truth, and falsity

$$\frac{}{(\Gamma) R(t)} \{R \subset (\Delta) \in \Sigma \text{ and } \vdash (\Gamma) t : \Delta\} \quad \frac{}{(\Gamma) \top} \{\vdash \Gamma\} \quad \frac{}{(\Gamma) \perp} \{\vdash \Gamma\}$$

conjunction, disjunction, and implication

$$\frac{(\Gamma) \zeta_0 \quad (\Gamma) \zeta_1}{(\Gamma) \zeta_0 \wedge \zeta_1} \quad \frac{(\Gamma) \zeta_0 \quad (\Gamma) \zeta_1}{(\Gamma) \zeta_0 \vee \zeta_1} \quad \frac{(\Gamma) \zeta_0 \quad (\Gamma) \zeta_1}{(\Gamma) \zeta_0 \rightarrow \zeta_1}$$

universal and existential quantification

$$\frac{(\Gamma, x : A) \psi}{(\Gamma) (\exists x : A) \psi} \quad \frac{(\Gamma, x : A) \psi}{(\Gamma) (\forall x : A) \psi}$$

Proposition 16

1. If u occurs free in ϕ in $\vdash (\Delta) \phi$, then $\vdash (\Delta) u : B_i$ for some i , where B_1, \dots, B_n is
2. If $\vdash (\Gamma) t : \Delta$ and $\vdash (\Delta) \phi$, then $\vdash (\Gamma) \phi[t/y]$.

Definition 19. *A sentence is a formula with the empty context. We usually omit the context when denoting a sentence. A sequent is a triple $(\Gamma) \Phi \Rightarrow P$ such that $(\Gamma) \Phi$ and $(\Gamma) P$ are finite multisets of formulas. The multiset $(\Gamma) \Phi$ is called the antecedent and $(\Gamma) P$ the succedent. A sequent is called intuitionistic if the succedent has at most one formula.*

Definition 20. *Sequents are derived over a logic system according to the following clauses, called the inference rules for classical logic.*

logical axiom, \perp elimination, and \top introduction

$$\overline{(\Gamma) \zeta \Rightarrow \zeta} \quad \overline{(\Gamma) \perp \Rightarrow} \quad \overline{(\Gamma) \Rightarrow \top}$$

weakening left and right, contraction left and right

$$\frac{(\Gamma) \Phi \Rightarrow P}{(\Gamma) \zeta, \Phi \Rightarrow P} \quad \frac{(\Gamma) \Phi \Rightarrow P}{(\Gamma) \Phi \Rightarrow P, \zeta} \quad \frac{(\Gamma) \zeta, \zeta, \Phi \Rightarrow P}{(\Gamma) \zeta, \Phi \Rightarrow P} \quad \frac{(\Gamma) \Phi \Rightarrow P, \zeta, \zeta}{(\Gamma) \Phi \Rightarrow P, \zeta}$$

substitution and cut

$$\frac{(\Delta) \Psi \Rightarrow \Theta}{(\Gamma) \Psi[r/y] \Rightarrow \Theta[s/y]} \{ \vdash (\Gamma) r = s : \Delta \} \quad \frac{(\Gamma) \Phi_0 \Rightarrow P_0, \zeta \quad (\Gamma) \zeta, \Phi_1 \Rightarrow P_1}{(\Gamma) \Phi_0, \Phi_1 \Rightarrow P_0, P_1}$$

\wedge elimination and introduction

$$\frac{(\Gamma) \zeta_i, \Phi \Rightarrow P}{(\Gamma) \zeta_0 \wedge \zeta_1, \Phi \Rightarrow P} \quad \frac{(\Gamma) \Phi \Rightarrow P, \zeta_0 \quad (\Gamma) \Phi \Rightarrow P, \zeta_1}{(\Gamma) \Phi \Rightarrow P, \zeta_0 \wedge \zeta_1}$$

\vee elimination and introduction

$$\frac{(\Gamma) \zeta_0, \Phi \Rightarrow P \quad (\Gamma) \zeta_1, \Phi \Rightarrow P}{(\Gamma) \zeta_0 \vee \zeta_1, \Phi \Rightarrow P} \quad \frac{(\Gamma) \Phi \Rightarrow P, \zeta_i}{(\Gamma) \Phi \Rightarrow P, \zeta_0 \vee \zeta_1}$$

\rightarrow elimination and introduction

$$\frac{(\Gamma) \Phi \Rightarrow P, \zeta_0 \quad (\Gamma) \zeta_1, \Phi \Rightarrow P}{(\Gamma) \zeta_0 \rightarrow \zeta_1, \Phi \Rightarrow P} \quad \frac{(\Gamma) \zeta_0, \Phi \Rightarrow P, \zeta_1}{(\Gamma) \Phi \Rightarrow P, \zeta_0 \rightarrow \zeta_1}$$

\exists elimination and introduction

$$\frac{(\Gamma, u : A) \psi[u/v], \Phi \Rightarrow P}{(\Gamma) (\exists v : A) \psi, \Phi \Rightarrow P} \quad \frac{(\Gamma) \Phi \Rightarrow P, \psi[t/v]}{(\Gamma) \Phi \Rightarrow P, (\exists v : A) \psi} \{ \vdash (\Gamma) t : A \}$$

\forall elimination and introduction

$$\frac{(\Gamma) \psi[t/v], \Phi \Rightarrow P}{(\Gamma) (\forall v : A) \psi, \Phi \Rightarrow P} \{ \vdash (\Gamma) t : A \} \quad \frac{(\Gamma, u : A) \Phi \Rightarrow P, \psi[u/v]}{(\Gamma) \Phi \Rightarrow P, (\forall v : A) \psi}$$

Definition 21. The inference rules for intuitionistic logic are the same as those for classical logic except that the sequents must be intuitionistic and the \rightarrow elimination clause is replaced by:

$$\frac{(\Gamma) \Phi_0 \Rightarrow \zeta \quad (\Gamma) \zeta, \Phi_1 \Rightarrow P}{(\Gamma) \Phi_0, \Phi_1 \Rightarrow P}.$$

Definition 22. If a sequent $(\Gamma) \Phi \Rightarrow P$ is derivable over Σ using the inference rules of classical logic, then we write $\vdash_{\Sigma} (\Gamma) \Phi \Rightarrow P$. If the sequent is intuitionistic and the derivation uses the inference rules of intuitionistic logic, then we write instead $\vdash_{\Sigma}^i (\Gamma) \Phi \Rightarrow P$. We omit the subscript Σ if no confusion arises.

Definition 23. A first order theory over Σ is a set of sentences over Σ , called the first order axioms of the theory. A sentence ζ is said to be derivable from the first order axioms in S , written $S \vdash \zeta$, if $\vdash (\Gamma) \Phi \Rightarrow \zeta$ such that the sentences in Φ are all first order axioms in S . We again use a superscript \vdash^i in case the derivation is intuitionistic.

5 Structures for Dependently Sorted Logic

Having presented the systems for intuitionistic and classical logic in the previous section, we now proceed with their semantics. The striking aspect of the following development is, perhaps, that it is only a slight generalisation of that for multisorted logic.

We present complete semantics for both classical and intuitionistic logic. The definition of classical structure is straightforward: given a structure for the theory of sorts only the declared predicate symbols remain to be interpreted.

Definition 24. *Let Σ be a logic system. A classical structure M for Σ is a structure for Σ together with an assignment of a subset R_M of $\llbracket \Delta \rrbracket_M$ to each predicate symbol declaration $R \subset (\Delta)$.*

The interpretation of arbitrary formulas is then given by appropriate subsets of the interpretation of their contexts and is defined as follows.

Definition 25. *The satisfaction relation \models_Σ on M between formulas $(\Gamma) \zeta$ and elements e of $\llbracket \Gamma \rrbracket_M$ is defined by induction on the structure of $(\Gamma) \zeta$ as follows.*

1. $M, e \models (\Gamma) R(t)$ if $\llbracket (\Gamma) t : \Delta \rrbracket_M(e) \in R_M$,
2. $M, e \models (\Gamma) \top$,
3. $M, e \not\models (\Gamma) \perp$,
4. $M, e \models (\Gamma) \zeta_0 \wedge \zeta_1$ if $M, e \models (\Gamma) \zeta_0$ and $M, e \models (\Gamma) \zeta_1$,
5. $M, e \models (\Gamma) \zeta_0 \vee \zeta_1$ if $M, e \models (\Gamma) \zeta_0$ or $M, e \models (\Gamma) \zeta_1$,
6. $M, e \models (\Gamma) (\exists x : A)\psi$ if $M, (e, e') \models (\Gamma, x : A) \psi$
for some $(e, e') \in \llbracket \Gamma, x : A \rrbracket_M$,
7. $M, e \models (\Gamma) \zeta_0 \rightarrow \zeta_1$ if $M, e \models (\Gamma) \zeta_0$ implies $M, e \models (\Gamma) \zeta_1$,
8. $M, e \models (\Gamma) (\forall x : A)\psi$ if $M, (e, e') \models (\Gamma, x : A) \psi$
for all $e' \in \llbracket (\Gamma) A \rrbracket_M(e)$.

Proposition 17 (Substitution lemma). *For every formula $(\Delta) \phi$, terms $(\Gamma) t : \Delta$, and $e \in \llbracket \Gamma \rrbracket_M$,*

$$M, e \models (\Gamma) \phi[t/y] \text{ if and only if } M, \llbracket (\Gamma) t : \Delta \rrbracket_M(e) \models (\Delta) \phi.$$

Proof. By induction on the structure of $\vdash (\Delta) \phi$.

Definition 26. *A formula $(\Gamma) \zeta$ is valid in M , written $M \models_\Sigma (\Gamma) \zeta$, if $M, e \models_\Sigma (\Gamma) \zeta$ for all $e \in \llbracket \Gamma \rrbracket_M$. The classical structure M is a classical model of a first order theory S over Σ , written $M \models_\Sigma S$, if M is an algebra for Σ and every first order axiom in S is valid in M . The formula is a consequence of S , written $S \models_\Sigma (\Gamma) \zeta$, if it is valid in every model of S . Again we omit the subscript Σ if no confusion arises.*

We claim the completeness of the above semantics for classical logic.

Proposition 18 (Completeness). *For any first order theory S , for any sentence ζ ,*

$$S \vdash \zeta \text{ if and only if } S \models \zeta.$$

Proof. The “only if” direction of the claim, soundness, is proved by induction on the derivation of ζ from the first order axioms in S . The classical approach to the other direction is to show that if ζ is not derivable from the first order axioms in S , then S has a model that does not satisfy ζ . The traditional proof which proceeds by first extending the theory to a complete Henkin one – a theory such that for any sentence, either it or its negation is derivable and such that there is a constant witnessing every derivable existential – and then by defining a classical structure interpreting the predicate symbols in the canonical algebra through derivability, like that in [16] for the case of enumerable languages or the more general one in [17], carries over to our case. Such a proof carried over to dependently sorted languages without equality may be found in [4].

We now proceed to the semantics for the intuitionistic systems. We define a notion of Kripke structure composed of classical structures and generalised in the sense that between the nodes we allow arbitrary classical structure morphisms. We should note that this generalisation is not essential to our results, as the standard definition, which only considers inclusions between the nodes, should suffice.

Definition 27. *Let N be a classical structure. A classical structure morphism on M to N , also called an extension of M , is a family h of functions indexed by the sorts over Σ such that*

1. *for all sort $(\Delta) D$, all $d \in \llbracket \Delta \rrbracket_M$, all $d' \in \llbracket (\Delta) D \rrbracket_M(d)$,*

$$h_{(\Delta) D}(d') \in \llbracket (\Delta) D \rrbracket_N(h_\Delta(d))$$

2. *for all term $(\Delta) p : A$, all $d \in \llbracket \Delta \rrbracket_M$,*

$$\llbracket (\Delta) p : A \rrbracket_N(h_\Delta(d)) = h_{(\Delta) A}(\llbracket (\Delta) p : A \rrbracket_M(d))$$

3. *for all $R \subset (\Delta) \in \Sigma$, all $d \in \llbracket \Delta \rrbracket_M$*

$$d \in R_M \text{ only if } h_\Delta(d) \in R_N$$

where h_Δ is the induced assignment on contexts:

$$\begin{aligned} h_\epsilon &= \text{id}_{\{\emptyset\}} \\ h_{\Gamma, v:A}(e, e') &= (h_\Gamma(e), h_{(\Gamma) A}(e')) \end{aligned}$$

for all $e \in \llbracket \Gamma \rrbracket_M$ and $e' \in \llbracket (\Gamma) A \rrbracket_M(e)$.

Definition 28. *A (generalised) Kripke structure for Σ is any category of classical structures for Σ and classical structure morphisms between them.*

The extension of a formula is defined in the usual way through forcing, except that we now have to consider arbitrary structure morphisms on each node for implications and universal quantifications.

Definition 29. The forcing relation \Vdash_{Σ} on a Kripke structure K between classical structures M in K , formulas $(\Gamma) \zeta$, and elements e of $\llbracket \Gamma \rrbracket_M$ is defined by induction on the structure of $(\Gamma) \zeta$ as follows.

1. $M, e \Vdash (\Gamma) R(t)$ if $\llbracket (\Gamma) t : \Delta \rrbracket_M(e) \in R_M$,
2. $M, e \Vdash (\Gamma) \top$,
3. $M, e \not\Vdash (\Gamma) \perp$,
4. $M, e \Vdash (\Gamma) \zeta_0 \wedge \zeta_1$ if $M, e \Vdash (\Gamma) \zeta_0$ and $M, e \Vdash (\Gamma) \zeta_1$
5. $M, e \Vdash (\Gamma) \zeta_0 \vee \zeta_1$ if $M, e \Vdash (\Gamma) \zeta_0$ or $M, e \Vdash (\Gamma) \zeta_1$,
6. $M, e \Vdash (\Gamma) (\exists x : A)\psi$ if $M, (e, e') \Vdash (\Gamma, x : A) \psi$
for some $(e, e') \in \llbracket \Gamma, x : A \rrbracket_M$,
7. $M, e \Vdash (\Gamma) \zeta_0 \rightarrow \zeta_1$ if $N, h_{\Gamma}(e) \Vdash (\Gamma) \zeta_0$ implies $N, h_{\Gamma}(e) \Vdash (\Gamma) \zeta_1$
for all classical structure N and classical structure morphism $h : M \rightarrow N$ in K ,
8. $M, e \Vdash (\Gamma) (\forall x : A)\psi$ if $N, (h_{\Gamma}(e), e') \Vdash (\Gamma, x : A) \psi$
for all $(h_{\Gamma}(e), e') \in \llbracket \Gamma, x : A \rrbracket_N$, classical structure N , and classical structure morphism $h : M \rightarrow N$ in K .

Definition 30. A formula $(\Gamma) \zeta$ is valid at a classical structure M in K , written $M \Vdash_{\Sigma} (\Gamma) \zeta$, if $M, e \Vdash (\Gamma) \zeta$ for all $e \in \llbracket \Gamma \rrbracket_M$. The formula is valid in K , written $K \Vdash_{\Sigma} (\Gamma) \zeta$, if $M \Vdash_{\Sigma} (\Gamma) \zeta$ for all classical structure M in K . The Kripke structure K is a Kripke model of a first order theory S , written $K \Vdash S$, if every first order axiom in S is valid in K . The formula is a Kripke consequence of S , written $S \Vdash_{\Sigma} (\Gamma) \zeta$, if it is valid in every Kripke model of S .

Proposition 19 (Substitution lemma). Let $(\Delta) \phi$ be a formula, let M be a classical structure in K , and let $e \in \llbracket \Gamma \rrbracket_M$. Suppose that $\vdash (\Gamma) t : \Delta$. Then

$$M, e \Vdash (\Gamma) \phi[t/y] \text{ if and only if } M, \llbracket (\Gamma) t : \Delta \rrbracket_M(e) \Vdash (\Delta) \phi.$$

Proof. By induction on the structure of $\vdash (\Delta) \phi$.

Proposition 20 (Completeness). For any first order theory S and sentence ζ

$$S \vdash^i \zeta \text{ if and only if } S \Vdash \zeta$$

Proof. The ‘‘only if’’ part of the claim is easy. For the other, one assumes a sentence ζ not derivable intuitionistically from the first order axioms in S and proceeds to build a Kripke model that does not force it. Roughly, the traditional proof considers all possible saturations – extensions of the theory such that for every derivable disjunction at least one of its disjuncts is derivable and such that there is a constant witnessing every derivable existential – of S and then takes as a model the category composed of the canonical structures of each of those saturations and of all classical structure morphisms between them. This can be carried over to our semantics. One can show that the model thus constructed does not force ζ from the fact that ζ is not derivable in at least one of the saturations.

Acknowledgments

This work was carried out under the supervision of Professor Peter Aczel. Also, many thanks to the referees for their generous and helpful comments.

References

1. Gambino, N., Aczel, P.: The generalised type-theoretic interpretation of constructive set theory (preprint 2005)
2. Jacobs, B.: *Categorical Logic and Type Theory. Studies in Logic and the Foundations of Mathematics*, vol. 141. North Holland, Amsterdam (1999)
3. Aczel, P.: Predicate logic with dependent sorts or types (unpublished 2004)
4. Belo, J.F.: *Dependently typed predicate logic*. Master's thesis, University of Manchester (2004)
5. Makkai, M.: *First order logic with dependent sorts, with applications to category theory* (unpublished, 1995)
6. Rabe, F.: First-order logic with dependent types. In: Furbach, U., Shankar, N. (eds.) *IJCAR 2006. LNCS (LNAI)*, vol. 4130, pp. 377–391. Springer, Heidelberg (2006)
7. Mosses, P.D. (ed.): *CASL Reference Manual. LNCS*, vol. 2960. Springer, Heidelberg (2004)
8. Benke, M., Dybjer, P., Jansson, P.: Universes for generic programs and proofs in dependent type theory. *Nordic J. of Computing* 10(4), 265–289 (2003)
9. Pfeifer, H., Rueß, H.: Polytypic proof construction. In: Bertot, Y., Dowek, G., Hirschowitz, A., Paulin, C., Théry, L. (eds.) *TPHOLs 1999. LNCS*, vol. 1690, pp. 55–72. Springer, Heidelberg (1999)
10. Cartmell, J.: *Generalized algebraic theories and contextual categories*. PhD thesis, Univ. Oxford (1978)
11. Cartmell, J.: Generalized algebraic theories and contextual categories. *Ann. Pure Appl. Logic* 32, 209–243 (1986)
12. Pitts, A.M.: Categorical logic. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) *Handbook of Logic in Computer Science. Algebraic and Logical Structures*, vol. 5, ch.2, Oxford University Press, Oxford (2000)
13. Hofmann, M.: Syntax and semantics of dependent types. In: Pitts, A.M., Dybjer, P. (eds.) *Semantics and Logics of Computation*, vol. 14, pp. 79–130. Cambridge University Press, Cambridge (1997)
14. Troelstra, A.S., Schwichtenberg, H.: *Basic proof theory*. Cambridge University Press, Cambridge (2000)
15. Johnstone, P.T.: *Sketches of an Elephant: A Topos Theory Compendium*, vol. 2. Oxford University Press, Oxford (2002)
16. Johnstone, P.T.: *Notes on Logic and Set Theory*. Cambridge University Press, Cambridge (1987)
17. Shoenfield, J.R.: *Mathematical Logic*. Association for Symbolic Logic (1967)